REMARKS

Claims 1-54 were originally filed in the present application, and claim 55 was subsequently added. No claims are currently amended, canceled or added. Consequently, claims 1-55 are currently pending in the present application. Reconsideration of the present application in light of the following remarks is respectfully requested.

Rejections under 35 U.S.C. §102

Claim 1

Claim 1 was rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent Publication No. 2001/0044327 of Kanefsky ("Kanefsky"). The PTO provides in MPEP §2131 that to anticipate a claim, the reference must teach every element of the claim. Therefore, to sustain this rejection, Kanefsky must disclose all of the elements of claim 1. However, as explained below, Kanefsky does not disclose all elements of claim 1.

Method for performing an operation on a hierarchical data tree

Although not in the same sense as in the present application, Kanefsky does disclose a hierarchical data tree. Nonetheless, Kanefsky clearly does not disclose performing an operation on the hierarchical data tree.

Kanefsky discloses visiting different portions of the data tree to display a series of nested messages to a user as the user navigates among a series of nested folders each associated with the messages. This is not the same as performing an operation on the data tree.

For example, the data in the data tree, such as the messages, folders, and folder contents, do not change. In the present application, the operation performed on the hierarchical data tree results in a change in the data of the data tree. However, no such change in the data is disclosed in Kanefsky. In contrast, the data in Kanefsky's tree is static – it does not change, whether in response to the performance of an operation thereon or otherwise. The data in Kanefsky's tree is

predetermined and pre-programmed; the user merely navigates through the data based upon selecting an icon in a folder, where such selection merely leads to another folder in the nested folder series.

An operation can be defined as an expression that derives a new value from one or more other values. An operator, such as the addition operator (+) or concatenation operator (&), determines how the new value is derived. An operation can similarly defined as any mathematical process, such as addition, subtraction, multiplication, division, raising to a power, etc. An operation can also be defined as a specification of a transformation of data. An operation is often a function performed on one or more values to produce one or more new values. The performance of an operation inherently implies a change in the object upon which the operation is performed. Examples of operations in the present application supporting these definitions include:

- "The specific number of neighboring nodes retrieved during the step 920 is dependent upon the particular operation desired to be performed on the data tree. For example, if the data tree represents a solid model of a silicon feature in a nano-electro-mechanical system (NEMS), micro-electro-mechanical system (MEMS) or other microelectronic device or physical structure, and if the operation being performed is an etch or other type of removal of a portion of the silicon feature, wherein the removed portion is to have a thickness of about 2 pixels, voxels or other element forming the elemental level nodes, two or three successively lower neighboring nodes may be retrieved (or the data representative thereof)." (Paragraph [0031]).
- "For example, if the operation being performed by the processing is to etch or otherwise remove a portion of the solid model represented by the hierarchical data tree, wherein the removed portion is to have a thickness of about 3 voxels, the processing may entail examining three lower neighboring nodes with each anchor node. In such an embodiment, the pre-operation status of the anchor node and the three lower neighbors

may be represented in binary form by four indicators as "0-1-1-1" wherein the anchor node is empty as represented by a "0" and the three lower neighboring nodes substantially comprise silicon as represented by a "1." After performing the operation to remove the top 3 voxels of silicon, the post-operation status of the anchor node and the three lower neighbors may be represented as "0-0-0-0." Thus, the key/value pair for this example may resemble "0-1-1-1/0-0-0-0," representing the pre- and post-operation configurations of the anchor node and the associated neighboring nodes." (Paragraph [0034]).

- "In other embodiments, the operation performed on the data tree may be a method of associating like components or finding connected components. In such embodiments, the operation may comprise assigning the anchor node to one of a plurality of equivalence classes, wherein each node in an equivalence class may have a common trait, characteristic or parameter, or may point to one or more common nodes, or may collectively form a single component. In such or similar embodiments, the operation performed on the anchor node may include maintaining a count of the number of nodes assigned to each equivalence class. The operation performed on the data tree may also comprise updating a parameter of the anchor node and/or neighboring nodes. For example, temperature data, pressure data or stress data may be updated to reflect a step in a fabrication process. In such an embodiment, the keys in the cache may represent node temperatures before the operation, and the values in the cache may represent node temperatures after the operation." (Paragraph [0035]).

Kanefsky fails to disclose anything resembling any of the above examples of operations disclosed in the present application. Kanefsky also fails to disclose anything falling within the possible definitions of an operation in the context of the present application.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses an operation that is performed on a hierarchical data tree. That is, the Examiner points to Kanefsky

paragraphs [0067]-[0074] as allegedly disclosing the performance of an operation on a hierarchical data tree. However, these paragraphs in Kanefsky merely disclose performing an action in response to the user's navigation of the data tree. They do not disclose performing an operation *on* the data tree. For example, Kanefsky teaches in Fig. 6 and paragraphs [0073]-[0074] that a script 628 can perform a first operation designed to attract a consumer's attention, and if the consumer responds to the script 628, the script 628 can evoke a second operation to provide the consumer with tire options and prices from which to choose, search the databases of vendors via the Internet to assure that a vendor has the consumer's tire choice in stock, reserve/purchase the tires using the consumer's credit card information, arrange for a towing, and place a phone call to the tire vendor. These operations ("a first operation" and "a second operation") are merely actions performed in response the consumer navigating the data tree, and are not operations in the context of the present application. That is, none of the operations taught in Kanefsky are performed on the data tree, and none of the operations change the data in Kanefsky's data tree. Thus, although Kanefsky uses the word "operation" to describe an action taken in response to user navigation of the data tree, it is not the same as the operation recited in the claims of the present application – it is not an operation performed *on* the data tree.

Put another way, Kanefsky teaches querying the data tree in response to the consumer's navigation of the data tree, then displaying the results of the query and subsequently performing an action based on or using the query results. However, the data resulting from the query is not subsequently changed. If the consumer repeats the same series of selections from the nested folders at a later point in time (e.g., the following day, week, or month), the same action will result from such navigation. In contrast, the present application is directed towards performing an operation *on* the data tree by querying the data tree, potentially changing the data resulting from the query by performing the operation, then replacing the data resulting from the query with the data resulting from the operation. Clearly, these represent two patentably distinct concepts. One concept (Kanefsky) involves using the data of a data tree to perform an action or service,

while the other concept (the present application) involves changing the data of a data tree by performing an operation on the data.

In view of the above, simple logic requires that Kanefsky also fails to disclose the elements of claim 1, as described below. That is, if Kanefsky fails to disclose the general concept of performing an operation on a hierarchical data tree, then it is logical that Kanefsky also fails to disclose the specific aspects of performing an operation on a hierarchical data tree.

Querying a pre-/post-operation data pair cache for a key representing the anchor node and the plurality of neighboring nodes in a pre-operation condition

Kanefsky does not disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Thus, necessarily, Kanefsky also fails to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition.

In contrast, Kanefsky merely discloses a series of nested folders, including a first level "root" folder and lower level "deck" folders, where the root and deck folders each contain a number of items which may include links to email messages, web pages, etc. This series of nested folders and the items contained therein are clearly not the same as, or even analogous to, the above-described pre-/post-operation data pair cache. For example, the series of nested folders and items therein do not include pairs of data representing both a pre-operation condition and post-operation condition. One reason for this is that Kanefsky fails to disclose performing an operation *on* the series of nested folders and items contained therein, as described in the previous section above. Moreover, because the folders are not the same as the pre-/post-operation data pair cache of the present application, Kanefsky also fails to disclose searching the folders and items therein for a key representing an anchor one of the folders/items and a plurality of neighboring folders/items in a pre-operation condition. That is, Kanefsky fails to disclose that

the folders/items have a pre-operation condition (or a post-operation condition). Accordingly, it is impossible that Kanefsky can disclose searching the folders/items for a key representing a pre-operation folder and its neighboring folders.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses a cache containing pre-/post-operation data pairs. In contrast, the Examiner has merely pointed to sections of Kanefsky which fail to mention a data cache of any kind.

Replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match

As described above, Kanefsky fails to disclose performing an operation *on* data in a hierarchical data tree. As also described above, Kanefsky also fails to disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Additionally, as described above, Kanefsky must also necessarily fail to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition. In view of all of this, it is undeniable that Kanefsky also fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match.

For example, Kanefsky fails to teach changing pre-operation data to post-operation data by performing an operation *on* the data, thereby generating post-operation data. Consequently, since Kanefsky fails to disclose generating post-operation data, Kanefsky must also fail to disclose replacing pre-operation data with generated post-operation data. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with generated post-operation data.

As also described above, Kanefsky fails to disclose querying a cache containing data pairs, where each pair includes pre-operation data and post-operation data representing

before/after configurations of an anchor node and its neighboring nodes. Consequently, since Kanefsky fails to disclose such cache querying, Kanefsky necessarily also fails to disclose replacing pre-operation data with data retrieved from such a cache. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with data retrieved from a cache.

In addition, Kanefsky necessarily fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match since Kanefsky fails to disclose: (1) generating post-operation data by performing an operation on pre-operation data; (2) the existence of a cache containing pre-/post-operation data pairs; and (3) querying such a cache. Each of these three elements, among others, is required to replace pre-operation data with cached post-operation data or generated post-operation data based on whether the cache query finds a match. Since Kanefsky fails to disclose any such elements, Kanefsky necessarily fails to disclose the claimed replacement of pre-operation data with post-operation data. Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match.

In view of any one or more of the reasons described above, the §102 rejection of claim 1 is not supported by Kanefsky. Therefore, Applicants respectfully request the Examiner withdraw the rejection.

Claim 27

Claim 27 was also rejected under 35 U.S.C. §102(b) as being anticipated by Kanefsky. As described above, sustaining this rejection requires that Kanefsky must disclose all of the elements of claim 27. However, as explained below, Kanefsky does not disclose all elements of claim 27.

System for performing an operation on a hierarchical data tree

Although not in the same sense as in the present application, Kanefsky does disclose a hierarchical data tree. Nonetheless, Kanefsky clearly does not disclose a system for performing an operation on the hierarchical data tree.

Kanefsky discloses visiting different portions of the data tree to display a series of nested messages to a user as the user navigates among a series of nested folders each associated with the messages. This is not the same as performing an operation on the data tree.

For example, the data in the data tree, such as the messages, folders, and folder contents, do not change. In the present application, the operation performed on the hierarchical data tree results in a change in the data of the data tree. However, no such change in the data is disclosed in Kanefsky. In contrast, the data in Kanefsky's tree is static – it does not change, whether in response to the performance of an operation thereon or otherwise. The data in Kanefsky's tree is predetermined and pre-programmed; the user merely navigates through the data based upon selecting an icon in a folder, where such selection merely leads to another folder in the nested folder series.

An operation can be defined as an expression that derives a new value from one or more other values. An operator, such as the addition operator (+) or concatenation operator (&), determines how the new value is derived. An operation can similarly defined as any mathematical process, such as addition, subtraction, multiplication, division, raising to a power, etc. An operation can also be defined as a specification of a transformation of data. An

operation is often a function performed on one or more values to produce one or more new values. The performance of an operation inherently implies a change in the object upon which the operation is performed. Examples of operations in the present application supporting these definitions include:

- "The specific number of neighboring nodes retrieved during the step 920 is dependent upon the particular operation desired to be performed on the data tree. For example, if the data tree represents a solid model of a silicon feature in a nano-electro-mechanical system (NEMS), micro-electro-mechanical system (MEMS) or other microelectronic device or physical structure, and if the operation being performed is an etch or other type of removal of a portion of the silicon feature, wherein the removed portion is to have a thickness of about 2 pixels, voxels or other element forming the elemental level nodes, two or three successively lower neighboring nodes may be retrieved (or the data representative thereof)." (Paragraph [0031]).
- "For example, if the operation being performed by the processing is to etch or otherwise remove a portion of the solid model represented by the hierarchical data tree, wherein the removed portion is to have a thickness of about 3 voxels, the processing may entail examining three lower neighboring nodes with each anchor node. In such an embodiment, the pre-operation status of the anchor node and the three lower neighbors may be represented in binary form by four indicators as "0-1-1-1" wherein the anchor node is empty as represented by a "0" and the three lower neighboring nodes substantially comprise silicon as represented by a "1." After performing the operation to remove the top 3 voxels of silicon, the post-operation status of the anchor node and the three lower neighbors may be represented as "0-0-0-0." Thus, the key/value pair for this example may resemble "0-1-1-1/0-0-0-0," representing the pre- and post-operation configurations of the anchor node and the associated neighboring nodes." (Paragraph [0034]).

- "In other embodiments, the operation performed on the data tree may be a method of associating like components or finding connected components. In such embodiments, the operation may comprise assigning the anchor node to one of a plurality of equivalence classes, wherein each node in an equivalence class may have a common trait, characteristic or parameter, or may point to one or more common nodes, or may collectively form a single component. In such or similar embodiments, the operation performed on the anchor node may include maintaining a count of the number of nodes assigned to each equivalence class. The operation performed on the data tree may also comprise updating a parameter of the anchor node and/or neighboring nodes. For example, temperature data, pressure data or stress data may be updated to reflect a step in a fabrication process. In such an embodiment, the keys in the cache may represent node temperatures before the operation, and the values in the cache may represent node temperatures after the operation." (Paragraph [0035]).

Kanefsky fails to disclose anything resembling any of the above examples of operations disclosed in the present application. Kanefsky also fails to disclose anything falling within the possible definitions of an operation in the context of the present application.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses an operation that is performed on a hierarchical data tree. That is, the Examiner points to Kanefsky paragraphs [0067]-[0074] as allegedly disclosing the performance of an operation on a hierarchical data tree. However, these paragraphs in Kanefsky merely disclose performing an action in response to the user's navigation of the data tree. They do not disclose performing an operation *on* the data tree. For example, Kanefsky teaches in Fig. 6 and paragraphs [0073]-[0074] that a script 628 can perform a first operation designed to attract a consumer's attention, and if the consumer responds to the script 628, the script 628 can evoke a second operation to provide the consumer with tire options and prices from which to choose, search the databases of

vendors via the Internet to assure that a vendor has the consumer's tire choice in stock, reserve/purchase the tires using the consumer's credit card information, arrange for a towing, and place a phone call to the tire vendor. These operations ("a first operation" and "a second operation") are merely actions performed in response the consumer navigating the data tree, and are not operations in the context of the present application. That is, none of the operations taught in Kanefsky are performed on the data tree, and none of the operations change the data in Kanefsky's data tree. Thus, although Kanefsky uses the word "operation" to describe an action taken in response to user navigation of the data tree, it is not the same as the operation recited in the claims of the present application – it is not an operation performed *on* the data tree.

Put another way, Kanefsky teaches querying the data tree in response to the consumer's navigation of the data tree, then displaying the results of the query and subsequently performing an action based on or using the query results. However, the data resulting from the query is not subsequently changed. If the consumer repeats the same series of selections from the nested folders at a later point in time (e.g., the following day, week, or month), the same action will result from such navigation. In contrast, the present application is directed towards performing an operation *on* the data tree by querying the data tree, potentially changing the data resulting from the query by performing the operation, then replacing the data resulting from the query with the data resulting from the operation. Clearly, these represent two patentably distinct concepts. One concept (Kanefsky) involves using the data of a data tree to perform an action or service, while the other concept (the present application) involves changing the data of a data tree by performing an operation on the data.

In view of the above, simple logic requires that Kanefsky also fails to disclose the elements of claim 1, as described below. That is, if Kanefsky fails to disclose the general concept of performing an operation on a hierarchical data tree, then it is logical that Kanefsky also fails to disclose the specific aspects of performing an operation on a hierarchical data tree.

Means for querying a pre-/post-operation data pair cache for a key representing the anchor node and the plurality of neighboring nodes in a pre-operation condition

Kanefsky does not disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Thus, necessarily, Kanefsky also fails to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition.

In contrast, Kanefsky merely discloses a series of nested folders, including a first level "root" folder and lower level "deck" folders, where the root and deck folders each contain a number of items which may include links to email messages, web pages, etc. This series of nested folders and the items contained therein are clearly not the same as, or even analogous to, the above-described pre-/post-operation data pair cache. For example, the series of nested folders and items therein do not include pairs of data representing both a pre-operation condition and post-operation condition. One reason for this is that Kanefsky fails to disclose performing an operation *on* the series of nested folders and items contained therein, as described in the previous section above. Moreover, because the folders are not the same as the pre-/post-operation data pair cache of the present application, Kanefsky also fails to disclose searching the folders and items therein for a key representing an anchor one of the folders/items and a plurality of neighboring folders/items in a pre-operation condition. That is, Kanefsky fails to disclose that the folders/items have a pre-operation condition (or a post-operation condition). Accordingly, it is impossible that Kanefsky can disclose searching the folders/items for a key representing a pre-operation folder and its neighboring folders.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses a cache containing pre-/post-operation data pairs. In contrast, the Examiner has merely pointed to sections of Kanefsky which fail to mention a data cache of any kind.

Means for replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match

As described above, Kanefsky fails to disclose performing an operation *on* data in a hierarchical data tree. As also described above, Kanefsky also fails to disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Additionally, as described above, Kanefsky must also necessarily fail to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition. In view of all of this, it is undeniable that Kanefsky also fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match.

For example, Kanefsky fails to teach changing pre-operation data to post-operation data by performing an operation *on* the data, thereby generating post-operation data. Consequently, since Kanefsky fails to disclose generating post-operation data, Kanefsky must also fail to disclose replacing pre-operation data with generated post-operation data. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with generated post-operation data.

As also described above, Kanefsky fails to disclose querying a cache containing data pairs, where each pair includes pre-operation data and post-operation data representing before/after configurations of an anchor node and its neighboring nodes. Consequently, since Kanefsky fails to disclose such cache querying, Kanefsky necessarily also fails to disclose replacing pre-operation data with data retrieved from such a cache. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with data retrieved from a cache.

In addition, Kanefsky necessarily fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache

query finds a match since Kanefsky fails to disclose: (1) generating post-operation data by performing an operation on pre-operation data; (2) the existence of a cache containing pre-/post-operation data pairs; and (3) querying such a cache. Each of these three elements, among others, is required to replace pre-operation data with cached post-operation data or generated post-operation data based on whether the cache query finds a match. Since Kanefsky fails to disclose any such elements, Kanefsky necessarily fails to disclose the claimed replacement of pre-operation data with post-operation data. Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match.

In view of any one or more of the reasons described above, the §102 rejection of claim 27 is not supported by Kanefsky. Therefore, Applicants respectfully request the Examiner withdraw the rejection.

Claim 39

Claim 39 was also rejected under 35 U.S.C. §102(b) as being anticipated by Kanefsky. As described above, sustaining this rejection requires that Kanefsky must disclose all of the elements of claim 39. However, as explained below, Kanefsky does not disclose all elements of claim 39.

Means for performing an operation on a hierarchical data tree

Although not in the same sense as in the present application, Kanefsky does disclose a hierarchical data tree. Nonetheless, Kanefsky clearly does not disclose a system for performing an operation on the hierarchical data tree.

Kanefsky discloses visiting different portions of the data tree to display a series of nested messages to a user as the user navigates among a series of nested folders each associated with the messages. This is not the same as performing an operation on the data tree.

For example, the data in the data tree, such as the messages, folders, and folder contents, do not change. In the present application, the operation performed on the hierarchical data tree results in a change in the data of the data tree. However, no such change in the data is disclosed in Kanefsky. In contrast, the data in Kanefsky's tree is static – it does not change, whether in response to the performance of an operation thereon or otherwise. The data in Kanefsky's tree is predetermined and pre-programmed; the user merely navigates through the data based upon selecting an icon in a folder, where such selection merely leads to another folder in the nested folder series.

An operation can be defined as an expression that derives a new value from one or more other values. An operator, such as the addition operator (+) or concatenation operator (&), determines how the new value is derived. An operation can similarly defined as any mathematical process, such as addition, subtraction, multiplication, division, raising to a power, etc. An operation can also be defined as a specification of a transformation of data. An operation is often a function performed on one or more values to produce one or more new values. The performance of an operation inherently implies a change in the object upon which the operation is performed. Examples of operations in the present application supporting these definitions include:

- "The specific number of neighboring nodes retrieved during the step 920 is dependent upon the particular operation desired to be performed on the data tree. For example, if the data tree represents a solid model of a silicon feature in a nano-electro-mechanical system (NEMS), micro-electro-mechanical system (MEMS) or other microelectronic device or physical structure, and if the operation being performed is an etch or other type of removal of a portion of the silicon feature, wherein the removed portion is to have a

thickness of about 2 pixels, voxels or other element forming the elemental level nodes, two or three successively lower neighboring nodes may be retrieved (or the data representative thereof)." (Paragraph [0031]).

- "For example, if the operation being performed by the processing is to etch or otherwise remove a portion of the solid model represented by the hierarchical data tree, wherein the removed portion is to have a thickness of about 3 voxels, the processing may entail examining three lower neighboring nodes with each anchor node. In such an embodiment, the pre-operation status of the anchor node and the three lower neighbors may be represented in binary form by four indicators as "0-1-1-1" wherein the anchor node is empty as represented by a "0" and the three lower neighboring nodes substantially comprise silicon as represented by a "1." After performing the operation to remove the top 3 voxels of silicon, the post-operation status of the anchor node and the three lower neighbors may be represented as "0-0-0-0." Thus, the key/value pair for this example may resemble "0-1-1-1/0-0-0-0," representing the pre- and post-operation configurations of the anchor node and the associated neighboring nodes." (Paragraph [0034]).
- "In other embodiments, the operation performed on the data tree may be a method of associating like components or finding connected components. In such embodiments, the operation may comprise assigning the anchor node to one of a plurality of equivalence classes, wherein each node in an equivalence class may have a common trait, characteristic or parameter, or may point to one or more common nodes, or may collectively form a single component. In such or similar embodiments, the operation performed on the anchor node may include maintaining a count of the number of nodes assigned to each equivalence class. The operation performed on the data tree may also comprise updating a parameter of the anchor node and/or neighboring nodes. For example, temperature data, pressure data or stress data may be updated to reflect a step in a fabrication process. In such an embodiment, the keys in the cache may represent node

temperatures before the operation, and the values in the cache may represent node temperatures after the operation." (Paragraph [0035]).

Kanefsky fails to disclose anything resembling any of the above examples of operations disclosed in the present application. Kanefsky also fails to disclose anything falling within the possible definitions of an operation in the context of the present application.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses an operation that is performed on a hierarchical data tree. That is, the Examiner points to Kanefsky paragraphs [0067]-[0074] as allegedly disclosing the performance of an operation on a hierarchical data tree. However, these paragraphs in Kanefsky merely disclose performing an action in response to the user's navigation of the data tree. They do not disclose performing an operation *on* the data tree. For example, Kanefsky teaches in Fig. 6 and paragraphs [0073]-[0074] that a script 628 can perform a first operation designed to attract a consumer's attention, and if the consumer responds to the script 628, the script 628 can evoke a second operation to provide the consumer with tire options and prices from which to choose, search the databases of vendors via the Internet to assure that a vendor has the consumer's tire choice in stock, reserve/purchase the tires using the consumer's credit card information, arrange for a towing, and place a phone call to the tire vendor. These operations ("a first operation" and "a second operation") are merely actions performed in response the consumer navigating the data tree, and are not operations in the context of the present application. That is, none of the operations taught in Kanefsky are performed on the data tree, and none of the operations change the data in Kanefsky's data tree. Thus, although Kanefsky uses the word "operation" to describe an action taken in response to user navigation of the data tree, it is not the same as the operation recited in the claims of the present application – it is not an operation performed *on* the data tree.

Put another way, Kanefsky teaches querying the data tree in response to the consumer's navigation of the data tree, then displaying the results of the query and subsequently performing

an action based on or using the query results. However, the data resulting from the query is not subsequently changed. If the consumer repeats the same series of selections from the nested folders at a later point in time (e.g., the following day, week, or month), the same action will result from such navigation. In contrast, the present application is directed towards performing an operation *on* the data tree by querying the data tree, potentially changing the data resulting from the query by performing the operation, then replacing the data resulting from the query with the data resulting from the operation. Clearly, these represent two patentably distinct concepts. One concept (Kanefsky) involves using the data of a data tree to perform an action or service, while the other concept (the present application) involves changing the data of a data tree by performing an operation on the data.

In view of the above, simple logic requires that Kanefsky also fails to disclose the elements of claim 1, as described below. That is, if Kanefsky fails to disclose the general concept of performing an operation on a hierarchical data tree, then it is logical that Kanefsky also fails to disclose the specific aspects of performing an operation on a hierarchical data tree.

Means for querying a pre-/post-operation data pair cache for a key representing the anchor node and the plurality of neighboring nodes in a pre-operation condition

Kanefsky does not disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Thus, necessarily, Kanefsky also fails to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition.

In contrast, Kanefsky merely discloses a series of nested folders, including a first level "root" folder and lower level "deck" folders, where the root and deck folders each contain a number of items which may include links to email messages, web pages, etc. This series of nested folders and the items contained therein are clearly not the same as, or even analogous to,

the above-described pre-/post-operation data pair cache. For example, the series of nested folders and items therein do not include pairs of data representing both a pre-operation condition and post-operation condition. One reason for this is that Kanefsky fails to disclose performing an operation *on* the series of nested folders and items contained therein, as described in the previous section above. Moreover, because the folders are not the same as the pre-/post-operation data pair cache of the present application, Kanefsky also fails to disclose searching the folders and items therein for a key representing an anchor one of the folders/items and a plurality of neighboring folders/items in a pre-operation condition. That is, Kanefsky fails to disclose that the folders/items have a pre-operation condition (or a post-operation condition). Accordingly, it is impossible that Kanefsky can disclose searching the folders/items for a key representing a pre-operation folder and its neighboring folders.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses a cache containing pre-/post-operation data pairs. In contrast, the Examiner has merely pointed to sections of Kanefsky which fail to mention a data cache of any kind.

Means for replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match

As described above, Kanefsky fails to disclose performing an operation *on* data in a hierarchical data tree. As also described above, Kanefsky also fails to disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Additionally, as described above, Kanefsky must also necessarily fail to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition. In view of all of this, it is undeniable that Kanefsky also fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match.

For example, Kanefsky fails to teach changing pre-operation data to post-operation data by performing an operation *on* the data, thereby generating post-operation data. Consequently, since Kanefsky fails to disclose generating post-operation data, Kanefsky must also fail to disclose replacing pre-operation data with generated post-operation data. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with generated post-operation data.

As also described above, Kanefsky fails to disclose querying a cache containing data pairs, where each pair includes pre-operation data and post-operation data representing before/after configurations of an anchor node and its neighboring nodes. Consequently, since Kanefsky fails to disclose such cache querying, Kanefsky necessarily also fails to disclose replacing pre-operation data with data retrieved from such a cache. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with data retrieved from a cache.

In addition, Kanefsky necessarily fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match since Kanefsky fails to disclose: (1) generating post-operation data by performing an operation on pre-operation data; (2) the existence of a cache containing pre-/post-operation data pairs; and (3) querying such a cache. Each of these three elements, among others, is required to replace pre-operation data with cached post-operation data or generated post-operation data based on whether the cache query finds a match. Since Kanefsky fails to disclose any such elements, Kanefsky necessarily fails to disclose the claimed replacement of pre-operation data with post-operation data. Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match.

In view of any one or more of the reasons described above, the §102 rejection of claim 39 is not supported by Kanefsky. Therefore, Applicants respectfully request the Examiner withdraw the rejection.

Claim 54

Claim 54 was also rejected under 35 U.S.C. §102(b) as being anticipated by Kanefsky. As described above, sustaining this rejection requires that Kanefsky must disclose all of the elements of claim 54. However, as explained below, Kanefsky does not disclose all elements of claim 54.

Method for performing an operation on a hierarchical data tree

Although not in the same sense as in the present application, Kanefsky does disclose a hierarchical data tree. Nonetheless, Kanefsky clearly does not disclose performing an operation on the hierarchical data tree.

Kanefsky discloses visiting different portions of the data tree to display a series of nested messages to a user as the user navigates among a series of nested folders each associated with the messages. This is not the same as performing an operation on the data tree.

For example, the data in the data tree, such as the messages, folders, and folder contents, do not change. In the present application, the operation performed on the hierarchical data tree results in a change in the data of the data tree. However, no such change in the data is disclosed in Kanefsky. In contrast, the data in Kanefsky's tree is static – it does not change, whether in response to the performance of an operation thereon or otherwise. The data in Kanefsky's tree is predetermined and pre-programmed; the user merely navigates through the data based upon selecting an icon in a folder, where such selection merely leads to another folder in the nested folder series.

An operation can be defined as an expression that derives a new value from one or more other values. An operator, such as the addition operator (+) or concatenation operator (&), determines how the new value is derived. An operation can similarly defined as any mathematical process, such as addition, subtraction, multiplication, division, raising to a power, etc. An operation can also be defined as a specification of a transformation of data. An operation is often a function performed on one or more values to produce one or more new values. The performance of an operation inherently implies a change in the object upon which the operation is performed. Examples of operations in the present application supporting these definitions include:

- "The specific number of neighboring nodes retrieved during the step 920 is dependent upon the particular operation desired to be performed on the data tree. For example, if the data tree represents a solid model of a silicon feature in a nano-electro-mechanical system (NEMS), micro-electro-mechanical system (MEMS) or other microelectronic device or physical structure, and if the operation being performed is an etch or other type of removal of a portion of the silicon feature, wherein the removed portion is to have a thickness of about 2 pixels, voxels or other element forming the elemental level nodes, two or three successively lower neighboring nodes may be retrieved (or the data representative thereof)." (Paragraph [0031]).
- "For example, if the operation being performed by the processing is to etch or otherwise remove a portion of the solid model represented by the hierarchical data tree, wherein the removed portion is to have a thickness of about 3 voxels, the processing may entail examining three lower neighboring nodes with each anchor node. In such an embodiment, the pre-operation status of the anchor node and the three lower neighbors may be represented in binary form by four indicators as "0-1-1-1" wherein the anchor node is empty as represented by a "0" and the three lower neighboring nodes substantially comprise silicon as represented by a "1." After performing the operation to remove the

top 3 voxels of silicon, the post-operation status of the anchor node and the three lower neighbors may be represented as "0-0-0-0." Thus, the key/value pair for this example may resemble "0-1-1-1/0-0-0-0," representing the pre- and post-operation configurations of the anchor node and the associated neighboring nodes." (Paragraph [0034]).

- "In other embodiments, the operation performed on the data tree may be a method of associating like components or finding connected components. In such embodiments, the operation may comprise assigning the anchor node to one of a plurality of equivalence classes, wherein each node in an equivalence class may have a common trait, characteristic or parameter, or may point to one or more common nodes, or may collectively form a single component. In such or similar embodiments, the operation performed on the anchor node may include maintaining a count of the number of nodes assigned to each equivalence class. The operation performed on the data tree may also comprise updating a parameter of the anchor node and/or neighboring nodes. For example, temperature data, pressure data or stress data may be updated to reflect a step in a fabrication process. In such an embodiment, the keys in the cache may represent node temperatures before the operation, and the values in the cache may represent node temperatures after the operation." (Paragraph [0035]).

Kanefsky fails to disclose anything resembling any of the above examples of operations disclosed in the present application. Kanefsky also fails to disclose anything falling within the possible definitions of an operation in the context of the present application.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses an operation that is performed on a hierarchical data tree. That is, the Examiner points to Kanefsky paragraphs [0067]-[0074] as allegedly disclosing the performance of an operation on a hierarchical data tree. However, these paragraphs in Kanefsky merely disclose performing an action in response to the user's navigation of the data tree. They do not disclose performing an

operation *on* the data tree. For example, Kanefsky teaches in Fig. 6 and paragraphs [0073]-[0074] that a script 628 can perform a first operation designed to attract a consumer's attention, and if the consumer responds to the script 628, the script 628 can evoke a second operation to provide the consumer with tire options and prices from which to choose, search the databases of vendors via the Internet to assure that a vendor has the consumer's tire choice in stock, reserve/purchase the tires using the consumer's credit card information, arrange for a towing, and place a phone call to the tire vendor. These operations ("a first operation" and "a second operation") are merely actions performed in response the consumer navigating the data tree, and are not operations in the context of the present application. That is, none of the operations taught in Kanefsky are performed on the data tree, and none of the operations change the data in Kanefsky's data tree. Thus, although Kanefsky uses the word "operation" to describe an action taken in response to user navigation of the data tree, it is not the same as the operation recited in the claims of the present application – it is not an operation performed *on* the data tree.

Put another way, Kanefsky teaches querying the data tree in response to the consumer's navigation of the data tree, then displaying the results of the query and subsequently performing an action based on or using the query results. However, the data resulting from the query is not subsequently changed. If the consumer repeats the same series of selections from the nested folders at a later point in time (e.g., the following day, week, or month), the same action will result from such navigation. In contrast, the present application is directed towards performing an operation *on* the data tree by querying the data tree, potentially changing the data resulting from the query by performing the operation, then replacing the data resulting from the query with the data resulting from the operation. Clearly, these represent two patentably distinct concepts. One concept (Kanefsky) involves using the data of a data tree to perform an action or service, while the other concept (the present application) involves changing the data of a data tree by performing an operation on the data.

In view of the above, simple logic requires that Kanefsky also fails to disclose the elements of claim 1, as described below. That is, if Kanefsky fails to disclose the general concept of performing an operation on a hierarchical data tree, then it is logical that Kanefsky also fails to disclose the specific aspects of performing an operation on a hierarchical data tree.

Querying a pre-/post-operation data pair cache for a key representing the anchor node and the plurality of neighboring nodes in a pre-operation condition

Kanefsky does not disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Thus, necessarily, Kanefsky also fails to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition.

In contrast, Kanefsky merely discloses a series of nested folders, including a first level "root" folder and lower level "deck" folders, where the root and deck folders each contain a number of items which may include links to email messages, web pages, etc. This series of nested folders and the items contained therein are clearly not the same as, or even analogous to, the above-described pre-/post-operation data pair cache. For example, the series of nested folders and items therein do not include pairs of data representing both a pre-operation condition and post-operation condition. One reason for this is that Kanefsky fails to disclose performing an operation *on* the series of nested folders and items contained therein, as described in the previous section above. Moreover, because the folders are not the same as the pre-/post-operation data pair cache of the present application, Kanefsky also fails to disclose searching the folders and items therein for a key representing an anchor one of the folders/items and a plurality of neighboring folders/items in a pre-operation condition. That is, Kanefsky fails to disclose that the folders/items have a pre-operation condition (or a post-operation condition). Accordingly, it

is impossible that Kanefsky can disclose searching the folders/items for a key representing a pre-operation folder and its neighboring folders.

Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses a cache containing pre-/post-operation data pairs. In contrast, the Examiner has merely pointed to sections of Kanefsky which fail to mention a data cache of any kind.

Replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match

As described above, Kanefsky fails to disclose performing an operation *on* data in a hierarchical data tree. As also described above, Kanefsky also fails to disclose a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes. Additionally, as described above, Kanefsky must also necessarily fail to disclose querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition. In view of all of this, it is undeniable that Kanefsky also fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match.

For example, Kanefsky fails to teach changing pre-operation data to post-operation data by performing an operation *on* the data, thereby generating post-operation data. Consequently, since Kanefsky fails to disclose generating post-operation data, Kanefsky must also fail to disclose replacing pre-operation data with generated post-operation data. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with generated post-operation data.

As also described above, Kanefsky fails to disclose querying a cache containing data pairs, where each pair includes pre-operation data and post-operation data representing before/after configurations of an anchor node and its neighboring nodes. Consequently, since

Kanefsky fails to disclose such cache querying, Kanefsky necessarily also fails to disclose replacing pre-operation data with data retrieved from such a cache. Moreover, the Examiner has failed to indicate with any specificity how Kanefsky teaches replacing pre-operation data with data retrieved from a cache.

In addition, Kanefsky necessarily fails to disclose replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match since Kanefsky fails to disclose: (1) generating post-operation data by performing an operation on pre-operation data; (2) the existence of a cache containing pre-/post-operation data pairs; and (3) querying such a cache. Each of these three elements, among others, is required to replace pre-operation data with cached post-operation data or generated post-operation data based on whether the cache query finds a match. Since Kanefsky fails to disclose any such elements, Kanefsky necessarily fails to disclose the claimed replacement of pre-operation data with post-operation data. Moreover, the Examiner has failed to specifically indicate how Kanefsky discloses replacing pre-operation data with cached post-operation data or generated post-operation data based on whether a data pair cache query finds a match.

In view of any one or more of the reasons described above, the §102 rejection of claim 1 is not supported by Kanefsky. Therefore, Applicants respectfully request the Examiner withdraw the rejection.

Rejections Under 35 U.S.C. §103: Kanefsky in view of Hsiung

Claims 7-10, 24-26, 36-38 and 48-50 were rejected under 35 U.S.C. §103 as being unpatentable over Kanefsky in view of U.S. Pat. No. 6,865,509 to Hsiung ("Hsiung"). Applicants traverse this rejection on the grounds that these references are defective in establishing a *prima facie* case of obviousness with respect to claims 1, 27 and 39.

As the PTO recognizes in MPEP §2142:

> *... The Examiner bears the initial burden of factually supporting any prima facie conclusion of obviousness. If the Examiner does not produce a prima facie case, the applicant is under no obligation to submit evidence of nonobviousness...*

It is submitted that, in the present case, the Examiner has not factually supported a *prima facie* case of obviousness for the following, mutually exclusive, reasons.

1. Even When Combined, the References Do Not Teach the Claimed Subject Matter

As provided in 35 U.S.C. §103:

> *A patent may not be obtained ... if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains ... (Emphasis added)*

Thus, when evaluating a claim for determining obviousness, all limitations of the claim must be evaluated. However, as described above, Kanefsky fails to teach each and every element of claims 1, 27 and 39. Moreover, Hsiung fails to cure Kanefsky's shortcomings because Hsiung possesses the same deficiencies as Kanefsky with respect to failing to disclose each and every element of claims 1, 27 and 39. For example, Hsiung fails to disclose performing an operation

on a hierarchical data tree; the existence of a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes; querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition; and replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match.

Therefore, because claims 7-10, 24-26, 36-38 and 48-50 depend from claims 1, 27 and 39, it is impossible for the combination of Kanefsky and Hsiung to render obvious the subject matter of any of claims 7-10, 24-26, 36-38 and 48-50, as a whole, and the explicit terms of §103 cannot be met. Accordingly, Applicants respectfully request the Examiner withdraw the rejection.

2. Nonanalogous art cannot be used to establish obviousness

The present application is directed towards solid model data manipulation, such as systems and methods for processing data stored in, for example, a directed acyclic graph octree. In contrast, Kanefsky is limited to methods and systems for allowing a wireless device user to navigate a menu whereby, in response to various navigation commands, a wireless server can retrieve extrinsic information based on the user's navigation. 35 USC §103 requires that obviousness be determined on the basis of whether at the time the invention was made a person of ordinary skill in the art to which the subject matter pertains would have found the claimed invention as a whole obvious. Although one of ordinary skill in the art is presumed to be aware of all prior art in the field to which the invention pertains, he is not presumed to be aware of prior art outside that field and the field of the problem to be solved, *i.e.*, nonanalogous art.

Accordingly, in assessing the propriety of any assertion of prior art as a basis for a *prima facie* case of obviousness, one must determine the scope or bounds of the knowledge of one of ordinary skill in the art, *i.e.*, the analogous art presumably known by one of ordinary skill in the

art. Here, Kanefsky is from a nonanalogous art, thus precluding any *prima facie* case of obviousness.

Moreover, the Patent Office classification of references are evidence of nonanalogy. (See MPEP § 2141.01(a)). However, the present application is in class 707 (data processing: database and file management or data structures), Kanefsky is in class 455 (telecommunications), and Hsiung is in class 702 (data processing, measuring, calibrating or testing).

Because Kanefsky is in class 455 (telecommunications), it is nonanalogous art with respect to the present application, which is in class 707 (data processing: database and file management or data structures). Thus, it is improper to use Kanefsky as a basis for a §103 rejection with respect to the claims of the present application.

Moreover, because Kanefsky is in class 455 (telecommunications), it is also nonanalogous art with respect to Hsiung, which is in class 702 (data processing, measuring, calibrating or testing). Thus, it is improper to combine Kanefsky and Hsiung as a basis for a §103 rejection with respect to the claims of any application.

Thus, for these independent reasons alone, the Examiner's burden of factually supporting a *prima facie* case of obviousness has clearly not been met. Therefore, Applicants respectfully request that the Examiner withdraw the rejections under 35 U.S.C. §103.

3. The Combination of References is Improper

Assuming, arguendo, that none of the above arguments for non-obviousness apply (which is clearly not the case based on the above), another mutually exclusive and compelling reason why Kanefsky and Hsiung cannot be applied to reject claims 1, 27 and 39, and their dependent claims, under 35 U.S.C. §103 is that neither Kanefsky nor Hsiung teaches, or even suggests, the desirability of combination as specified above and as claimed in claims 1, 27 and 39. That is, neither reference teaches or suggests, among other elements of claims 1, 27 and 39:

- performing an operation *on* a hierarchical data tree;

- querying a pre-operation/post-operation data pair cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition;
- replacing retrieved pre-operation data with cached post-operation data;
- performing an operation *on* retrieved pre-operation data to generate post-operation data;
- replacing retrieved pre-operation retrieved data with generated post-operation data; or
- storing generated post-operation data in a cache with associated retrieved pre-operation data.

Moreover, Kanefsky and Hsiung are nonanalogous art relative to each other. That is, Kanefsky is directed towards methods and systems for allowing a wireless device user to navigate a menu whereby, in response to various navigation commands, a wireless server can retrieve extrinsic information based on the user's navigation. In contrast, Hsiung is directed towards techniques for monitoring and/or controlling processes by comparing the current state of a first process to current, historical, and/or predicted state of the first process or a second process using statistical, structural, or physical models. Consequently, one skilled in the art pertinent to Kanefsky would not be motivated to look to the disclosure of Hsiung, whether to arrive at a combination germane to the present application or otherwise. Likewise, one skilled in the art pertinent to Hsiung would not be motivated to look to the disclosure of Kanefsky, whether to arrive at a combination germane to the present application or otherwise.

Thus, it is clear that neither Kanefsky nor Hsiung provides any incentive or motivation supporting the desirability of their combination. Therefore, there is simply no basis in the art for combining the references to support a 35 U.S.C. §103 rejection.

In this context, the courts have repeatedly held that obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching, suggestion or incentive supporting the combination. In the present case it is clear that the combination of Kanefsky and Hsiung can only arise from hindsight based on the present

application, because there exists no showing, suggestion, incentive or motivation in either reference for the combination as applied to claims 1, 27 or 39. Therefore, for this mutually exclusive reason, the Examiner's burden of factually supporting a *prima facie* case of obviousness has clearly not been met. Accordingly, Applicants respectfully request that the Examiner withdraw the rejection under 35 U.S.C. §103 based on Kanefsky and Hsiung.

Rejections Under 35 U.S.C. §103: Kanefsky in view of Schreiber

Claims 15-20 were rejected under 35 U.S.C. §103 as being unpatentable over Kanefsky in view of U.S. Pat. Publication No. 2002/0138353 of Schreiber. Applicants traverse this rejection on the grounds that these references are defective in establishing a *prima facie* case of obviousness with respect to claim 1.

As the PTO recognizes in MPEP §2142:

> *... The Examiner bears the initial burden of factually supporting any prima facie conclusion of obviousness. If the Examiner does not produce a prima facie case, the applicant is under no obligation to submit evidence of nonobviousness...*

It is submitted that, in the present case, the Examiner has not factually supported a *prima facie* case of obviousness for the following, mutually exclusive, reasons.

1. Even When Combined, the References Do Not Teach the Claimed Subject Matter

As provided in 35 U.S.C. §103:

> *A patent may not be obtained ... if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains ... (Emphasis added)*

Thus, when evaluating a claim for determining obviousness, all limitations of the claim must be evaluated. However, as described above, Kanefsky fails to teach each and every element of claim 1. Moreover, Schreiber fails to cure Kanefsky's shortcomings because Schreiber also fails to teach each and every element of claim 1. For example, Schreiber fails to disclose performing an operation on a hierarchical data tree; the existence of a cache containing data pairs, where each pair includes pre-operation data and post-operation data, and where each pair corresponds to an anchor node and a plurality of neighboring nodes; querying such a cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition; and replacing pre-operation data with cached post-operation data or generated post-operation data depending on whether the query finds a match.

Therefore, because claims 15-20 depend from claim 1, it is impossible for the combination of Kanefsky and Schreiber to render obvious the subject matter of any of claims 15-20, as a whole, and the explicit terms of §103 cannot be met. Accordingly, Applicants respectfully request the Examiner withdraw the rejection.

2. Nonanalogous art cannot be used to establish obviousness

The present application is directed towards solid model data manipulation, such as systems and methods for processing data stored in, for example, a directed acyclic graph octree. In contrast, Kanefsky is limited to methods and systems for allowing a wireless device user to navigate a menu whereby, in response to various navigation commands, a wireless server can retrieve extrinsic information based on the user's navigation. 35 USC §103 requires that obviousness be determined on the basis of whether at the time the invention was made a person of ordinary skill in the art to which the subject matter pertains would have found the claimed invention as a whole obvious. Although one of ordinary skill in the art is presumed to be aware

of all prior art in the field to which the invention pertains, he is not presumed to be aware of prior art outside that field and the field of the problem to be solved, *i.e.*, nonanalogous art.

Accordingly, in assessing the propriety of any assertion of prior art as a basis for a *prima facie* case of obviousness, one must determine the scope or bounds of the knowledge of one of ordinary skill in the art, *i.e.*, the analogous art presumably known by one of ordinary skill in the art. Here, Kanefsky is from a nonanalogous art, thus precluding any *prima facie* case of obviousness.

Moreover, the Patent Office classification of references are evidence of nonanalogy. (See MPEP § 2141.01(a)). However, the present application is in class 707 (data processing: database and file management or data structures), Kanefsky is in class 455 (telecommunications), and Schreiber is in class 705 (data processing: financial, business practice, management, or cost/price determination).

Because Kanefsky is in class 455 (telecommunications), it is nonanalogous art with respect to the present application, which is in class 707 (data processing: database and file management or data structures). Thus, it is improper to use Kanefsky as a basis for a §103 rejection with respect to the claims of the present application.

Moreover, because Kanefsky is in class 455 (telecommunications), it is also nonanalogous art with respect to Schreiber, which is in class 705 (data processing: financial, business practice, management, or cost/price determination). Thus, it is improper to combine Kanefsky and Schreiber as a basis for a §103 rejection with respect to the claims of any application.

Thus, for these independent reasons alone, the Examiner's burden of factually supporting a *prima facie* case of obviousness has clearly not been met. Therefore, Applicants respectfully request that the Examiner withdraw the rejections under 35 U.S.C. §103.

3. The Combination of References is Improper

Assuming, arguendo, that none of the above arguments for non-obviousness apply (which is clearly not the case based on the above), another mutually exclusive and compelling reason why Kanefsky and Schreiber cannot be applied to reject claim 1, and its dependent claims, under 35 U.S.C. §103 is that neither Kanefsky nor Schreiber teaches, or even suggests, the desirability of combination as specified above and as claimed in claim 1. That is, neither reference teaches or suggests, among other elements of claim 1:

- performing an operation *on* a hierarchical data tree;
- querying a pre-operation/post-operation data pair cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition;
- replacing retrieved pre-operation data with cached post-operation data;
- performing an operation *on* retrieved pre-operation data to generate post-operation data;
- replacing retrieved pre-operation retrieved data with generated post-operation data; or
- storing generated post-operation data in a cache with associated retrieved pre-operation data.

Moreover, Kanefsky and Schreiber are nonanalogous art relative to each other. That is, Kanefsky is directed towards methods and systems for allowing a wireless device user to navigate a menu whereby, in response to various navigation commands, a wireless server can retrieve extrinsic information based on the user's navigation. In contrast, Schreiber is directed towards databases that store records having fields with sets rather than single values therein. Consequently, one skilled in the art pertinent to Kanefsky would not be motivated to look to the disclosure of Schreiber, whether to arrive at a combination germane to the present application or otherwise. Likewise, one skilled in the art pertinent to Schreiber would not be motivated to look to the disclosure of Kanefsky, whether to arrive at a combination germane to the present application or otherwise.

Thus, it is clear that neither Kanefsky nor Schreiber provides any incentive or motivation supporting the desirability of their combination. Therefore, there is simply no basis in the art for combining the references to support a 35 U.S.C. §103 rejection.

In this context, the courts have repeatedly held that obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching, suggestion or incentive supporting the combination. In the present case it is clear that the combination of Kanefsky and Schreiber can only arise from hindsight based on the present application, because there exists no showing, suggestion, incentive or motivation in either reference for the combination as applied to claim 1. Therefore, for this mutually exclusive reason, the Examiner's burden of factually supporting a *prima facie* case of obviousness has clearly not been met. Accordingly, Applicants respectfully request that the Examiner withdraw the rejection under 35 U.S.C. §103 based on Kanefsky and Schreiber.

Rejections Under 35 U.S.C. §103: Kanefsky in view of Schreiber and Hsiung

Claims 11-14 were rejected under 35 U.S.C. §103 as being unpatentable over Kanefsky in view of Schreiber and Hsiung. Applicants traverse this rejection on the grounds that these references are defective in establishing a *prima facie* case of obviousness with respect to claim 1.

As the PTO recognizes in MPEP §2142:

> *... The Examiner bears the initial burden of factually supporting any prima facie conclusion of obviousness. If the Examiner does not produce a prima facie case, the applicant is under no obligation to submit evidence of nonobviousness...*

It is submitted that, in the present case, the Examiner has not factually supported a *prima facie* case of obviousness for the following, mutually exclusive, reasons.

1. Even When Combined, the References Do Not Teach the Claimed Subject Matter

As provided in 35 U.S.C. §103:

> *A patent may not be obtained ... if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains ... (Emphasis added)*

Thus, when evaluating a claim for determining obviousness, all limitations of the claim must be evaluated. However, as described above, Kanefsky fails to teach each and every element of claim 1. Moreover, as described above, Schreiber and Hsiung fail to cure Kanefsky's shortcomings because Schreiber and Hsiung each also fail to teach each and every element of claim 1. Therefore, because claims 11-14 depend from claim 1, it is impossible for the combination of Kanefsky, Schreiber and Hsiung to render obvious the subject matter of any of claims 11-14, as a whole, and the explicit terms of §103 cannot be met. Accordingly, Applicants respectfully request the Examiner withdraw the rejection.

2. Nonanalogous art cannot be used to establish obviousness

As described above, Kanefsky is nonanalogous art with respect to each of the present application, Hsiung and Schreiber. 35 USC §103 requires that obviousness be determined on the basis of whether at the time the invention was made a person of ordinary skill in the art to which the subject matter pertains would have found the claimed invention as a whole obvious. Although one of ordinary skill in the art is presumed to be aware of all prior art in the field to which the invention pertains, he is not presumed to be aware of prior art outside that field and the field of the problem to be solved, *i.e.*, nonanalogous art. Accordingly, for this independent reason alone, the Examiner's burden of factually supporting a *prima facie* case of obviousness

clearly cannot be met by any combination that includes Kanefsky. Therefore, Applicants respectfully request that the Examiner withdraw the rejections under 35 U.S.C. §103.

3. The Combination of References is Improper

Assuming, arguendo, that none of the above arguments for non-obviousness apply (which is clearly not the case based on the above), another, mutually exclusive, and compelling reason why Kanefsky, Schreiber and Hsiung cannot be applied to reject claim 1, and its dependent claims, under 35 U.S.C. §103 is that neither Kanefsky, Schreiber nor Hsiung teaches, or even suggests, the desirability of combination as specified above and as claimed in claim 1. That is, neither reference teaches or suggests, among other elements of claim 1:

- performing an operation *on* a hierarchical data tree;
- querying a pre-operation/post-operation data pair cache for a key representing an anchor node and a plurality of neighboring nodes in a pre-operation condition;
- replacing retrieved pre-operation data with cached post-operation data;
- performing an operation *on* retrieved pre-operation data to generate post-operation data;
- replacing retrieved pre-operation retrieved data with generated post-operation data; or
- storing generated post-operation data in a cache with associated retrieved pre-operation data.

Moreover, Schreiber and Hsiung are each nonanalogous art relative to Kanefsky. That is, Kanefsky is directed towards methods and systems for allowing a wireless device user to navigate a menu whereby, in response to various navigation commands, a wireless server can retrieve extrinsic information based on the user's navigation. In contrast, Schreiber is directed towards databases that store records having fields with sets rather than single values therein, and Hsiung is directed towards techniques for monitoring and/or controlling processes by comparing the current state of a first process to current, historical, and/or predicted state of the first process

or a second process using statistical, structural, or physical models. Consequently, one skilled in the art pertinent to Kanefsky would not be motivated to look to the disclosure of either Schreiber or Hsiung, whether to arrive at a combination germane to the present application or otherwise. Likewise, one skilled in the art pertinent to either Schreiber or Hsiung would not be motivated to look to the disclosure of Kanefsky, whether to arrive at a combination germane to the present application or otherwise.

Thus, it is clear that neither Kanefsky, Schreiber nor Hsiung provides any incentive or motivation supporting the desirability of their combination. Therefore, there is simply no basis in the art for combining the references to support a 35 U.S.C. §103 rejection.

In this context, the courts have repeatedly held that obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching, suggestion or incentive supporting the combination. In the present case it is clear that the combination of Kanefsky, Schreiber and Hsiung can only arise from hindsight based on the present application, because there exists no showing, suggestion, incentive or motivation in any of the references for the combination as applied to claim 1. Therefore, for this mutually exclusive reason, the Examiner's burden of factually supporting a *prima facie* case of obviousness has clearly not been met. Accordingly, Applicants respectfully request that the Examiner withdraw the rejection under 35 U.S.C. §103.

Examiner's Response to Applicants' Response to Non-Final Office Action

The Examiner responded to Applicants' previous explanation of Kanefsky being nonanalogous art by explaining that it has been held that a prior art reference must either be in the field of applicant's endeavors or, if not, then be reasonably pertinent to the particular problem with which the application was concerned, in order to be relied upon as a basis for rejection of the claimed invention. However, neither of these conditions are satisfied in the present scenario.

For example, Kanefsky is not in the field of Applicants' endeavors. Kanefsky is in the field of telecommunications. In direct contrast, Applicants' endeavors are directed towards solid model data manipulation, such as systems and methods for processing data stored in, for example, a directed acyclic graph octree. Clearly, Kanefsky's telecommunications teachings are not in the field of solid model manipulation or other forms of data processing, database and file management, or data structures. Thus, for this reason alone, Kanefsky is nonanalogous art.

Similarly, Kanefsky is not reasonably pertinent to the particular problem with which the present application is concerned. The field of Kanefsky's teachings (telecommunications) are not reasonably pertinent to solid model data manipulation, systems and methods for processing data stored in a directed acyclic graph octree, performing equivalence class merging with a particularly large and/or complex engineered or solid model, or the desire to perform such data manipulation on common desktop computers. These examples of the particular problems addressed by the present application are provided in the Background section of the application. However, none of these example problems are reasonably pertinent to Kanefsky's teachings regarding telecommunications. Thus, for this reason alone, Kanefsky is nonanalogous art.

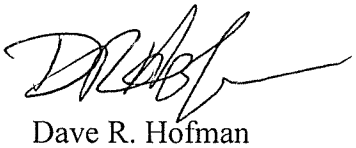
Consequently, Kanefsky cannot be relied upon as a basis for rejection of the claims of the present application, because Kanefsky is nonanalogous art with respect to the present application and with respect to Schreiber and Hsiung.

Conclusion

It is clear from all of the foregoing that independent claims 1, 27, 39 and 54 are in condition for allowance. Dependent claims 2-26, 28-38, 40-53 and 55 depend from and further limit independent claims 1, 27, 39 and 54 and, therefore, are allowable as well.

Should the Examiner deem that an interview with Applicants' undersigned attorney would expedite consideration, the Examiner is invited to call the undersigned attorney at the telephone number indicated below.

Respectfully submitted,



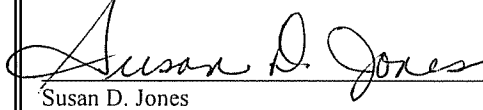
Dave R. Hofman
Registration No. 55,272

Dated: November 10, 2006

HAYNES AND BOONE, LLP
901 Main Street, Suite 3100
Dallas, Texas 75202-3789
Telephone: 972/739-8630
Facsimile: 214/200-0853
Attorney Docket No.: 34003.83
Document No.: H-642672.1

Certificate of Service

I hereby certify that this correspondence is being filed with the U.S. Patent and Trademark Office via EFS-Web on June 30, 2006.



Susan D. Jones